HW/SW 통합설계를 위한 웹 기반의 SpecCharts 기술 환경

김승권[†] · 김종훈[††]

요 약

본 논문에서는 분산 구조를 갖는 웹 기반 HW/SW 통합설계 환경(WebCEDA)을 제안하고, HW/SW 통합설계 절차 중에서 시스템 기술에 사용할 SpecCharts 기술 환경(ScSE)을 구현한다. WebCEDA는 기존의 통합설계 도구가 갖는 플랫폼 제한, 새로운 기술 추가의 어려움, 공동 작업 환경의 부재와 같은 단점을 해결할 수 있는 3층 클라이언트/서버 구조를 가진다. ScSE는 웹 인터페이스, SpecCharts 에디터, HW/SW 통합설계 응용 서버, SpecCharts 변환기로 구성된다. 다양한 예제를 SpecCharts 에디터 상에서 기술하고, SpecCharts 변환기를 통해 VHDL로 변환시킨 후 Synopsys 상에서 시뮬레이션을 수행함으로써 구현된 ScSE가 정확하게 동작함을 확인하였다. 위의 실험 결과를 통해 ScSE는 시스템 기술의 복잡도 감소, 시스템 설계자의 자연스런 추상화된 설계 유도 등의 장점이 있음을 보였다.

Web-based SpecCharts Specification Environment for HW/SW Codesign

Seung Kwon Kim[†] and Jong Hoon Kim[††]

ABSTRACT

In this paper, we propose a Web-based HW/SW Codesign Environment with Distributed Architecture (WebCEDA), then design and implement SpecCharts Specification Environment(ScSE) for specifying systems in WebCEDA. WebCEDA has 3-tier client/server architecture that can remedy disadvantages of existing codesign tools, such as platform dependency, difficulty of extension, absence of collaboration environment. ScSE includes web interface, SpecCharts editor, HW/SW codesign application server and SpecCharts translator. To verify the operation of ScSE, we specify several example system using SpecCharts editor, then translate it to VHDL using SpecCharts translator and simulate the translated VHDL codes on synopsys. As the results, we know that ScSE has correct operations, also obtain the following advantages, the reduction in system complexity and the natural abstract design.

1. 서 론

HW/SW 통합설계란 하드웨어와 소프트웨어로 구성된 시스템을 동시협조 설계하는 통합된 설계 패러다임을 말한다[1]. VLSI/CAD 기술의 발달로 디지털 시스템의 복잡도가 증가하고 시스템의 출시 기간이 점점 단축되면서 하드웨어와 소프트웨어의 통합적인 설계를 위한 설계 방법론에 대한 관심이 증대되고 있다. HW/SW 통합설계는 설계 절차를 가속화하여 제품의 출시 기간을 줄여주며, 하드웨어와 소프트웨어의 구현 비율을 동적으로 조절(Trade-off)할 수 있어서 성능상의 제약 조건을 만족하는 작은 비용의 제품을 제작하는 최적 설계를 가능하게 하는 장점이 있다. 현재 발표된 HW/SW 통합설계 도구는 Polis [2], CoDe-X, COOL[3], PeaCE[4], Chinook[5], Code Sign[6], SpecSyn[7] 등이 있다. 표 1은 기존의 통합

이 논문은 1997년도 동아대학교 학술연구조성비(공모과제)에 의하여 연구되었음

† 준회원, 동아대학교 컴퓨터공학과 대학원 박사과정
†† 종신회원, 동아대학교 전기전자컴퓨터공학부 교수

표 1. 기존의 HW/SW 통합설계 도구 분석

툴의 종류	구현 대상	시스템 기술 언어	내부 모델	사용 환경
Polis	내장형 시스템	Esterel	확장 FSM	단일 플랫폼
CoDe-X	Xputer	C	Flow Graph	단일 플랫폼
COOL	데이터 플로우 기반 시스템	VHDL 부분집합	State Transition Graph	단일 플랫폼
PeaCE	이질적 디지털 시스템	DFG+확장 FSM	DFG+확장 FSM	단일 플랫폼
Chinook	내장형 시스템	ACT 구성 모델	ACT 구성 모델	다중 플랫폼
CodeSign	복잡한 실시간 내장형 시스템	이질적 혼합 모델	FunState	다중 플랫폼
SpecSyn	내장형 시스템	SpecCharts	CDFG	단일 플랫폼

설계 도구를 구현 대상, 시스템 기술 언어, 내부 모델, 사용 환경을 기준으로 비교 · 분석한 결과이다.

각 HW/SW 통합설계 도구는 구현하고자 하는 대상 시스템의 차이로 인해 서로 다른 시스템 기술 언어와 내부 모델을 사용하고 있으며, 빠른 기술 발전과 사용자 요구에 의해 빠른 속도로 업그레이드되고 있어 생명 주기가 매우 짧다. 그럼에도 불구하고 새 버전의 통합설계 도구를 다운로드 받아서 다시 설치해야 하는 부담감으로 인해, 시스템 설계자가 발전된 통합설계 기법을 실제로 사용하기까지는 오랜 시간이 걸린다. 각 도구의 사용 플랫폼은 대부분이 스팍 머신과 Unix 운영체제로 제한되어 있으며, CodeSign과 Chinook의 최근 버전인 ipChinook의 경우만이 자바 기술을 이용하여 다중 플랫폼을 지원하지만 각 플랫폼에서의 설치 방법과 필요한 소프트웨어 구성 요소가 달라서 설치와 유지 보수에 많은 비용이 든다. 또한 기존의 통합설계 도구는 시스템 설계자의 협동 작업에 대한 고려가 전혀 되지 않고 있다.

이런 문제점을 해결하기 위해서는 개방성, 보안성, 공동 작업 환경, 기술 추가의 용이성, 쉬운 인터페이스를 갖춘 새로운 HW/SW 통합설계 도구의 개발이 필요하다. 설계 절차를 보다 생산성 있게 하기 위해서는 다양한 환경의 시스템 설계자들이 모두 접근할 수 있는 공통 인터페이스가 필요하며, 내부 모델의 변경과 같은 내부적인 기술 변화를 감출 수 있어야 한다. 또한 팀 단위의 설계와 자원 공유에 대한 수단이 고려되어야 한다. 최근 CAD/EDA 분야에서도 이런 문제점을 인식하여 Weld와 같은 통합된 웹-기반 환경의 개발에 착수하였다[8,9]. HW/SW 통합설계 도구는 개발 방법론의 다양성 때문에 내부 절차를 추상화할 필요가 있으며, 빠른 기술 발전으로 인해 생명 주기가 짧아 효과적으로 도구를 갱신할 수 있는 방법이 절실히 요구된다. 본 논문에서는 이런 문제점을 해결하기 위한 방안으로 분산 구조를 갖는 웹-기반 HW/SW 통합설계 환경인 WebCEDA(Web-based Codesign Environment with Distributed Architecture)를 제안하고, HW/SW 통합설계 절차상의 시스템 기술 단계를 WebCEDA 상에서 수행하기 위한 ScSE (SpecCharts Specification Environment)를 구현한다. 웹에 기반한 최초의 HW/SW 통합설계 환경인 WebCEDA는 지리적으로 분산된 설계자에게 효율적으로 설계 작업을 수행할 수 있는 공동 작업장을 제공한고, 또한 향상된 기술을 사용자 인터페이스의 변화를 최소화하면서 빠른 시간 내에 제안한 통합설계 환경에 적용할 수 있다. 시스템 설계자는 상위 수준에서 문제 자체에 초점을 맞추어 시스템을 설계할 수 있으며, 제안한 환경의 개방성으로 인해 시스템 설계자가 사용하고 있는 기존 환경을 이용할 수 있기 때문에, 추가적인 시스템 구입 등의 비용을 절감할 수 있다는 장점이 있다.

본 논문의 구성은 다음과 같다. 2절에서는 WebCEDA의 시스템 기술 언어인 SpecCharts 에 관해 설명한다. 3절에서 WebCEDA의 전체 구성과 ScSE의 역할에 대해 설명하고, 4절에서는 ScSE의 구성과 구현 환경 및 방법에 대해 기술하고, 5절에서는 ScSE의 성능 평가를 위해 사용된 예제와 이를 이용한 실험 결과를 보여준다. 마지막으로 6절에서는 결론과 향후 연구 과제를 제시하였다.

2. SpecCharts(PSM+VHDL)

2.1 시스템 기술 언어의 비교 · 분석

HW/SW 통합설계는 시스템 기술, 하드웨어/소프트웨어 분할, 합성, 통합 시뮬레이션 등의 과정을 거치게 되는데, 설계 절차를 자동화하기 위해서는 이들 중에서도 시스템을 기술하는 과정이 매우 중요하다.

현재 국내에서는 내장형 시스템의 설계를 위한 HW/SW 통합설계 방법이 연구되고 있는데, 시스템 기술 언어로서는 표준 언어인 VHDL과 C 언어를 함께 사용하고 있다[10]. 그러나 VHDL과 C 언어는 내장형 시스템을 기술하기 위해서 반드시 필요한 상태 전환이나 예외 처리 등의 기능을 기술하지 못하며, 계층성에 있어서도 완전한 기술이 어렵기 때문에 내장형 시스템을 설계하기 위한 시스템 기술 언어로 사용되기에는 부적합하다. 그럼에도 불구하고 VHDL과 C를 시스템 기술 언어로 사용하는 이유는 이를 대체할 적합한 언어가 없었기 때문이다. 현재 HW/SW 통합설계를 위한 시스템 기술 언어로서는 Ptolemy와 같은 이질적 기술[11], Esterel[12]이나 LOTOS[13]와 같은 특정 목적 언어가 고려되고 있다. 그러나 이들 언어는 범용성이 부족할 뿐만 아니라, HW/SW 통합설계 절차상의 합성 작업을 수행하기 위해서는 그 언어를 지원하는 신뢰성 있는 합성 툴을 개발하거나, 합성 툴이 존재하는 기존의 언어로 변환 작업을 수행해야 하는데, 그러한 작업을 수행하기가 매우 까다롭다. 또한 현재 가장 많이 사용되고 있는 VHDL과 C는 모두 실행 가능한 시스템 기술 언어로서, 이미 많은 설계자들에게 익숙해져 있고 합성 작업을 지원하는 다양한 도구들이 나와 있기 때문에, 이를 대체할 수 있는 새로운 시스템 기술 언어의 개발은 시간적으로나 경제적으로 비능률적이다.

표 2는 WebCEDA의 시스템 기술 언어를 결정하기 위해서, 내장형 시스템의 설계시 필요한 특성에 대한 각 시스템 기술 언어의 지원 여부를 분석한 결과이다. 이를 통해서 HW/SW 통합설계 기법을 이용한 내장형 시스템의 설계에는 SpecCharts와 Hawk가 최적의 시스템 기술 언어로 고려될 수 있음을 알 수 있다.

표 2. 내장형 시스템의 개념적 모델 특성을 위한 언어 지원

시스템 기술 언어	내장형 시스템의 설계에 필요한 특성들				
	상태 전환	계층성	병렬성	프로그래밍	예외 처리
VHDL	×	▲	●	●	×
Verilog	×	●	●	●	●
HardwareC	×	▲	●	●	×
Statecharts	●	●	●	×	●
SDL	●	▲	●	×	×
Esterel	×	●	●	●	●
SpecCharts	●	●	●	●	●
Hawk	●	●	●	●	●

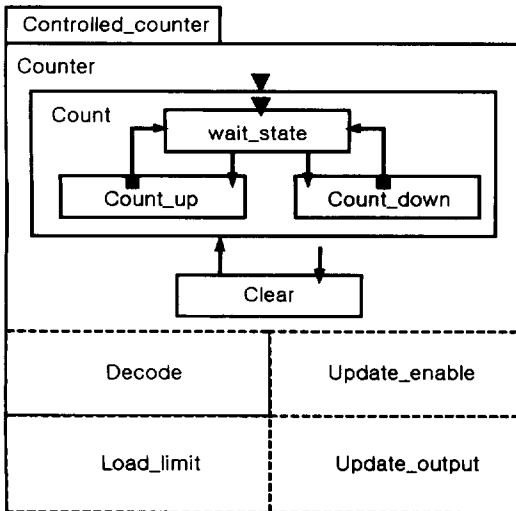
●: 완전히 지원함,
▲: 부분적으로 지원함
×: 전혀 지원하지 않음

Esterel은 검증 기법을 통해서 시스템의 동작을 확인할 수 있는 장점이 있는 반면 시뮬레이션이 어려운 단점이 있다. 이를 극복하기 위해 Haskell에 기반한 Hawk가 개발되었지만, 널리 알려져 있지 못할 뿐 아니라 시스템 설계자에게 익숙하지 않은 함수형 패러다임의 프로그래밍 기법을 이해해야 하는 단점이 있다. SpecCharts는 VHDL에 계층적·병렬적 기능을 갖는 상태 다이어그램을 결합시킴으로써, VHDL이 가지지 못한 상태전환, 계층성, 예외 처리 등의 기능이 새롭게 추가, 확장된 시스템 기술 언어이다[14]. SpecCharts는 기본적으로 시스템 설계자에게 널리 알려진 VHDL을 기본 모델로 설계되었기 때문에 학습 부담을 최소화할 수 있다. 또한 SpecCharts를 VHDL로 변환하는 알고리즘이 알려져 있기 때문에, VHDL을 기반으로 진행된 기존 연구를 그대로 수행할 수 있는 이점이 있다. 이러한 이유로 WebCEDA의 시스템 기술 언어로 SpecCharts를 채택하였다.

2.2 SpecCharts를 이용한 시스템 설계 과정

SpecCharts를 이용한 시스템 설계 과정은 전체적인 시스템을 기본 객체인 Behavior를 계층적으로 구성함으로써 시작한다. 계층적 구조를 갖는 각각의 Behavior들은 내부적으로 또 다른 Behavior 또는 VHDL 문장을 포함하며, 전이 아크(Transition Arc)를 통해서 상태 전환이 이루어진다. Behavior는 순차적으로 실행되는 하위 Behavior를 포함하는 Sequential Subbehavior, 병렬적으로 수행되는 하위 Behavior를 포함하는 Concurrent Subbehavior, 그 자체로써 하나의 VHDL 문장이 되는 Code로 나뉘어진다. 전이 아크는 하위 Behavior의 동작이 완료되는 시점에 전이가 발생하는 TOC(Transaction-On-Completion)와 하위 Behavior의 동작에 관계없이 특정 이벤트의 발생에 따라 전이가 발생하는 TI(Transaction-Immediately)로 나뉜다. 각각은 <T, C, NB>의 형식으로 표현되는데, T는 전이의 종류를 나타내며, C는 전이가 발생하는 상태를 뜻한다. NB는 Next Behav-

ior의 약자로 전이의 발생으로 인해 어떤 Behavior로 상태가 전이될 것인지를 나타낸다. SpecCharts에서의 동기화는 VHDL에서와 같이 wait 문을 이용해서 이벤트의 발생을 검사함으로써 이루어지며, TI 아크를 이용함으로써 예외상황에 대한 처리를 할 수 있다.

SpecCharts는 그래픽 표현과 텍스트 표현이라는 2가지 모드로 시스템을 기술할 수 있다. 그림 1은 SpecCharts를 이용해 다양한 제어가 가능한 카운터를 설계한 예이다.

```
Entity Controlled_counter is
port(CLK, STRB : in bit;
   CON : in bit_vector(1 downto 0);
   DATA : in nibble;  CNT_OUT : out nibble);
end Controlled_counter;

Architecture aCC is
  signal incc : boolean := false;
  .......
  behavior Counter type sequential subbehavior is
  begin
    wait_state : (TI, CONSIG[2] = '1' and EN and
rising(CLK), Count_up);
      wait_state : (TI, CONSIG[3] = '1' and EN and
rising(CLK), Count_down);
      Count_up : (TOC, true, wait_state);
      Count_down : (TOC, true, wait_state);

      behavior Count_down type code is
      begin
        CNT <= SUB(CNT, B"0001") after 12 ns;
      ........
```

그림 1. SpecCharts를 이용한 시스템 기술 예 : Controlled Counter

3. WebCEDA

3.1 WebCEDA의 구조

기존의 HW/SW 통합설계 도구의 문제점을 해결하기 위해 제안한 WebCEDA는 그림 2와 같은 3층 클라이언트/서버 구조를 가진다. HW/SW 통합설계 도구는 많은 응용 프로그램 서비스를 제공하며, 구현을 위해 C++, Java, Perl 등과 같이 서로 다른 프로그램 언어가 사용된다. 또한 빠른 기술 변화로 인해 많은 변경과 추가가 예상되며, 시스템 설계시 발생하는 대규모 트랜잭션을 효율적으로 처리하기 위한 수단이 고려되어야 한다. 또한 공동 작업시 동일 데이터베이스를 접근하는 경우가 많다. 전통적인 2층 클라이언트/서버 환경으로는 위와 같은 요구 조건을 만족하기 어렵기 때문에, 클라이언트와 HW/SW 통합설계 서버, 리모트 서버로 이루어진 3층 클라이언트/서버 구조가 되어야 한다. 클라이언트는 사용자 인터페이스를 담당하는 부분으로 동적 웹 페이지로 구성된 사용자 인터페이스와 자바 빈/Active X/CORBA 객체로 구성되어 설계 절차 관리, 시스템 기술, 시각화(Visualization) 등을 담당하는 클라이언트 프로그램이다. HW/SW 통합설계 서버는 응용 프로그램 논리(Application Logic)를 담당하는 부분으로, 웹 서버, 트랜잭션 서버, 응용 서버, 보안 서버로 구성된다. 리모트 서버는 백엔드 자원 관리(Backend Resource Manage)를 담당하는 부분으로 HTML 저장소, 문서 저장소, 파일 시스템, 데이터베이스, 통합설계에 사용되는 응용 프로그램으로 구성된다.

WebCEDA는 기존의 HW/SW 통합설계 도구의 문제점을 세션 서비스를 이용한 협동 작업 환경의 제공, 객체 지향 사용자 인터페이스(OOUI)를 이용한

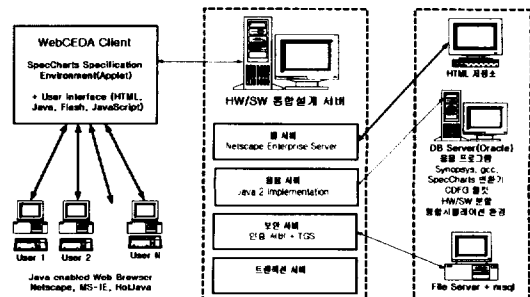



그림 2. WebCEDA의 구조

사용자 인터페이스 변경의 최소화, 트랜잭션 관리를 통한 효율적인 설계 데이터의 전송, 보안 서버를 통한 안정성 확보 등의 기능으로 해결할 수 있다.

3.2 WebCEDA의 통합설계 절차

WebCEDA를 이용한 HW/SW 통합설계는 다음과 같은 순서로 이루어진다. ① 웹-기반 사용자 인터페이스에서 로그인 절차를 통해서 보안 서버와 통신함으로써, WebCEDA의 사용 권한을 얻는다. ② 원하는 시스템을 자바 애플릿 형태의 사용자 인터페이스인 ScSE 내에서 SpecCharts 언어를 통해 기술한다. ③ 자바 애플릿이 가지는 보안 제약을 극복하기 위한 파일 처리 기능을 가지는 HW/SW 통합설계 응용 서버는 시스템 설계자의 저장하기, 불러오기, 명령 실행 등의 요구를 처리한다. 애플릿과 HW/SW 통합설계 응용 서버 사이의 통신은 효율성과 애플릿 보안 제약을 극복하기 위해 소켓 인터페이스를 통해 구현하였다. 그러므로 HW/SW 통합설계 응용 서버의 내부 설계가 RPC, CORBA 등으로 변경되더라도 동일한 인터페이스를 제공할 수 있다. ④ SpecCharts의 그래픽 표현은 SpecCharts 에디터 내에 내장된 변환기를 통해 동일한 의미를 갖는 텍스트 표현으로 바뀌며, 그것은 다시 SpecCharts 변환기[15]를 통해 Synopsys에서 합성 가능한 VHDL 기술로 변환된다. Synopsys의 vhdlan과 vhdldbx를 이용해서 생성된 VHDL에 대한 논리적 오류 검증 및 시뮬레이션을 수행한다. ⑤ SpecCharts 변환기에 의해 생성된 VHDL은 적절한 검증 절차를 마치고 나면, VDT의 VHDL Analyzer[16]를 이용해서 HW/SW 자동 분할을 위한 내부 모델인 CDFG(Control Data Flow Graph)로 바뀐다. ⑥ HW/SW 분할 알고리즘[17]은 CDFG를 하드웨어 요소와 소프트웨어 요소로 분할하며 인터페이스 기능을 추가하여 다시 VHDL과 C로 변환된다. ⑦ 하드웨어와 소프트웨어로 나뉘어진 시스템은 GNU C Compiler와 Synopsys를 통해 합성되며, 통합 시뮬레이션 시스템[18]을 이용하여 성능 평가 및 전체 시스템의 기능 검증을 수행함으로써 완성된다. 그림 3은 WebCEDA를 이용한 HW/SW 통합설계 절차를 도식화한 것이다.

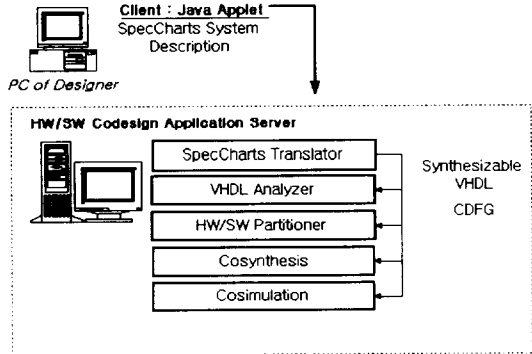



그림 3. WebCEDA를 이용한 통합설계 흐름

4. SpecCharts 기술 환경(ScSE)

SpecCharts 기술 환경은 웹 인터페이스, SpecCharts 에디터, HW/SW 통합설계 응용 서버, SpecCharts 변환기로 구성된다. ScSE는 웹 상에서 SpecCharts를 이용해 시스템을 기술하고, 그 결과를 VHDL로 변환시켜 HW/SW 분할에 필요한 형태로 바꿔주는 역할을 수행한다. 웹 인터페이스는 로그인 절차를 통해 사용자 세션을 관리하고, ScSE의 설계 절차를 관리하는 기능을 한다. SpecCharts 에디터에서 설계된 결과는 HW/SW 통합설계 응용 서버를 거쳐 SpecCharts 변환기에 전달된다. SpecCharts 변환기는 SpecCharts를 VHDL로 변환시켜 Synopsys와 같은 기존의 합성툴에서 시뮬레이션을 수행함으로써 설계 데이터의 오류를 검증할 수 있게 한다. 그림 4는 ScSE의 웹 인터페이스이다.

그림 4. ScSE의 웹 인터페이스

4.1 SpecCharts 에디터

현재 SpecCharts를 지원하는 SpecSyn과 같은 통

합 설계 관련 툴이 개발 중에 있지만, 플랫폼이 제한적이며, 웹을 기반으로 하는 본 연구에는 적합하지 않다. 이러한 이유로 본 연구에서는 SpecCharts의 그래픽 표현을 사용해서 시스템을 기술할 수 있는 자바 애플릿을 개발하였다. SpecCharts 에디터는 SpecCharts의 그래픽 모드와 텍스트 모드의 설계를 모두 지원하며, 기본적으로 그래픽 모드를 우선적으로 이용하도록 되어 있어 시스템 설계자로 하여금 상향식 접근 방식으로 시스템을 설계하도록 유도하였다. SpecCharts 에디터는 Solaris 2.7.1에서 JDK 1.2.1 버전을 이용해 개발하였으며, IE 4.0에서 테스트하였다.

SpecCharts 에디터를 구현하는데 있어 가장 문제가 되었던 애플릿의 보안 문제로 인한 기능 제약을 해결하기 위해, 객체 직렬화를 이용한 지속적 객체 저장 기법과 소켓을 이용한 애플릿 상에서의 클라이언트-서버 통신 기법을 사용하였다. 객체 직렬화는 실행시간 객체를 위한 저장 방식으로 플랫폼에 무관하게 설계 자료를 저장할 수 있다. 이 기능은 간단한 인터페이스 상속만으로 기능 구현이 가능한데, 원격지의 저장 객체를 활용할 수 있는 지속적 저장 메커니즘을 제공한다. SpecCharts 에디터는 직관적이며 세련된 인터페이스를 구현하기 위해, JDK 1.1의 후기 버전부터 지원된 인터페이스 구성 라이브러리인 스윙을 이용하였다. 이를 통해 플랫폼에 종속되지 않는 안정된 룩 엔 필(Look and Feel)을 지원하였다.

그림 5는 그림 1에서 소개한 Controlled Counter 예제를 SpecCharts 에디터를 통해 기술한 것이다. 각 behavior는 그 속성에 따라 서로 다른 색으로 표현되어 설계자는 직관적으로 어떠한 behavior인지 파악할 수 있다. Concurrent subbehavior들의 경계에 표시된 점선은 병렬성을 나타낸다. TI 아크는 화살표, TOC 아크는 끝에 종료점이 달려 있는 화살표로 표시된다. 각 전이 아크는 전이가 발생되는 이벤트를 표시함으로써 전체 명세를 쉽게 파악할 수 있도록 하였다. Sequential Subbehavior 중 시작이 되는 behavior는 역삼각형으로 표시되어 있다. SpecCharts의 그래픽 표현은 전체 Behavior의 계층성, 예외 처리, 상태 전환을 체계적으로 보여주지만, 프로그래밍 코드를 직접 각 Behavior에 나타내게 되면 전체 화면이 오히려 복잡해진다. 이 문제를 해결하기 위해, 각 Behavior나 전이 아크를 더블 클릭 하거나 좌측의 툴바에 있는 아이콘을 이용해 코드 윈도우를 띄우는

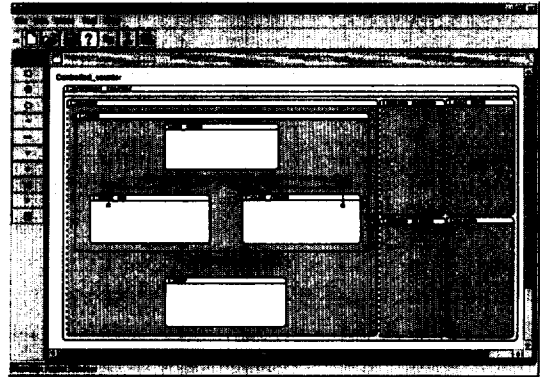



그림 5. SpecCharts 에디터의 인터페이스

방법을 채택하였다. 입력된 각 Behavior나 전이 아크는 속성 윈도우를 통해 크기나 위치 정보를 변경할 수 있으며, 마우스를 이용해서 객체를 드래그 앤 드롭해서 조작할 수도 있다.

SpecCharts를 이용한 시스템 기술은 대부분이 각 Behavior의 코드를 입력하는 것이다. 그림 6은 Behavior의 코드 입력 과정을 보인다. Behavior의 선언부와 동작 기술부를 분리하여 설계자의 편의성을 고려하였다. 전이 아크는 그 이름과 전이가 발생할 조건을 기술하도록 하였으며, 전이 아크의 종류와 전이의 출발점과 도착이 되는 Behavior는 자동으로 입력되도록 하였다.

그림 7은 전이 아크의 코드를 입력하는 과정을 보인 것이다. SpecCharts 에디터는 디스플레이의 한계

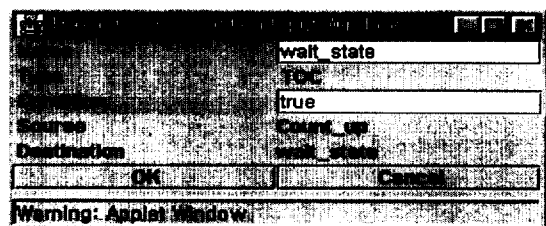
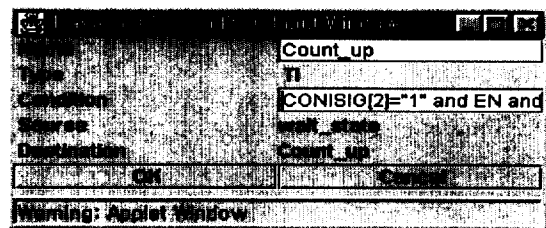



그림 6. 전이 아크(TOC, TI)의 속성 기술

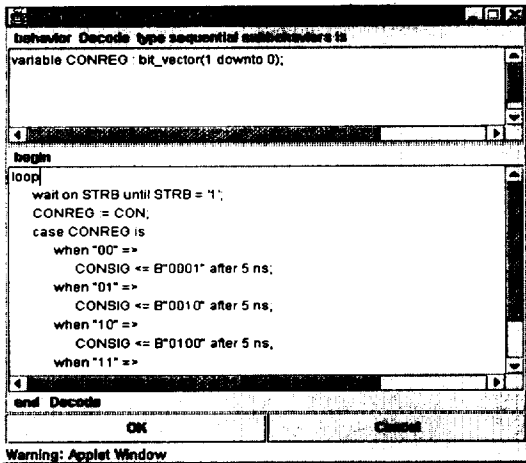



그림 7. Behavior 코드 입력 창

를 극복하기 위해 하위 Behavior 만의 설계로 확장하는 다이브(Dive) 기능, SpecCharts의 그래픽 표현을 텍스트 표현으로 바꾸는 기능, HW/SW 통합설계 응용 서버와의 네트워 연결 및 신뢰성 있는 파일 전송 기능을 포함시켰다. SpecCharts 에디터를 통해 기술된 코드는 객체 직렬화 기법을 이용해서, 소켓 통신으로 서버에 기록된다. 그러나 이 파일은 바이너리 파일로 다른 프로그램에서는 이용될 수 없다. SpecCharts 변환기의 입력으로 사용하기 위해서는 내장된 변환기를 통해 SpecCharts의 그래픽 표현을 텍스트 표현으로 변환시켜야 한다. 그림 8은 동일한 회로를 텍스트 표현으로 변환시킨 결과이다.

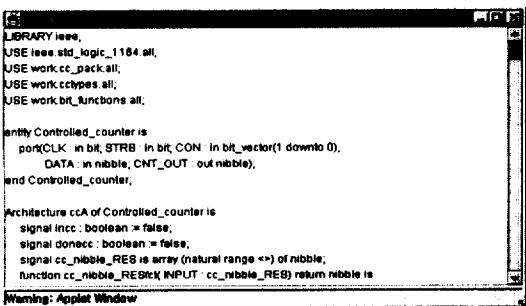



그림 8. 자동 생성된 SpecCharts의 텍스트 표현

4.2 HW/SW 통합설계 응용 서버

HW/SW 통합설계 응용 서버는 SpecCharts 에디터에서 설계된 자료를 서버에 저장하고, 설계자의 저장된 자료에 대한 요청을 처리하며, 서버에 있는 관련 응용 프로그램을 실행하고, 그 결과를 브라우저로 보내는 역할을 수행한다.

그림 9는 HW/SW 통합설계 응용 서버의 클래스 계층도를 UML로 나타낸 것이다.

HW/SW 통합설계 응용 서버의 동작 과정은 다음과 같다. 서버는 포트 번호를 이용해서 서버 소켓을 생성하고, 포트에 접근하는 클라이언트가 있을 때까지 대기한다. 매번 포트에 대한 접근 응답이 있을 경우, 클라이언트 핸들러를 생성한다. 핸들러는 클라이언트 소켓을 생성하며, 클라이언트 소켓으로부터 입출력 스트림을 생성한다. 클라이언트 소켓으로부터 메시지를 수신하고, 수신된 메시지를 통해 시스템 설계자의 명령을 판별한다. 명령이 쓰기인 경우, 다음 소켓으로부터 수신된 내용을 서버 측에 저장한다. 읽기의 경우, 서버 측에 쓰여진 내용을 소켓으로 전송하고, 실행인 경우에는 서버에 있는 명령을 실행하고 그 결과를 클라이언트에게 돌려준다.

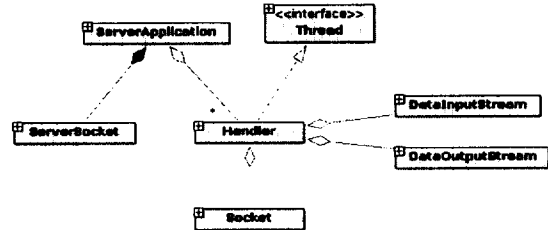



그림 9. HW/SW 통합설계 응용 서버의 클래스 계층도

4.3 SpecCharts 변환기

SpecCharts를 WebCEDA의 시스템 기술 언어로 사용하기 위해서는 SpecCharts 언어를 합성할 수 있는 합성툴을 제작해야 한다. 그러나 SpecCharts 언어를 위한 전용 합성툴을 설계하는데는 많은 비용이 들어가기 때문에, 같은 의미를 가지는 VHDL 언어로 변환시켜 주는 변환기를 설계해서 기존의 VHDL 합성툴들을 이용하는 것이 더 효과적이다.

SpecCharts를 VHDL로 변환하는 기본 알고리즘은 [9]에서 소개되었다. SpecCharts 변환 알고리즘에 의해 생성된 VHDL 문이 시스템 설계자가 의도한 것과 같은 의미를 갖도록 하기 위해, Synopsys 환경에 적합한 VHDL 코드를 생성하는 SpecCharts 변환기를 Solaris 2.5 플랫폼에서 Flex와 Bison을 사용하여 구현하였다.

SpecCharts 변환기의 입력은 SpecCharts 프로그램이 되고, 그 출력은 동일한 의미를 갖는 VHDL 문이다. SpecCharts에서 문법적으로 VHDL과 비교되는 점은 Behavior 문과 전이 아크가 추가되었다는 점이다. 전체적인 SpecCharts 기술은 크게 Entity와 Architecture로 나뉘는데, Entity 선언은 VHDL과 같다. Architecture 선언 내부만이 전이 아크로 묶여진 계층화된 Behavior로 이루어진다는 점에서 VHDL과 차이가 있을 뿐이다. 따라서, SpecCharts 변환기는 SpecCharts내의 Behavior문과 전이 아크를 통한 상태전환, 계층성, 예외 처리 기능을 동일한 기능의 VHDL 코드로 바꾸어주는 일을 한다고 말할 수 있다. 전체적인 알고리즘은 하나의 Behavior를 식별할 때마다 다시 변환 알고리즘을 적용하는 재귀적 방법에 의해 구성되어 있다. SpecCharts의 Behavior 중에서 Code는 그 자체가 VHDL 문장으로써, 특별한 변환 과정을 거치지 않는다.

SpecCharts 변환기를 설계하면서 가장 먼저 고려해야한 점은 복합된 Behavior를 VHDL로 표현하는 방법이다. SpecCharts의 계층성을 VHDL 상에서 표현할 수 있는 유일한 방법은 중첩된 블럭(Nested Block)문을 사용하는 것이다. Concurrent subbehavior는 하위 Behavior들이 서로 병렬적으로 수행되어야 하기 때문에 각각을 process문으로 변환시킨다. 상태 전환 기능을 VHDL을 통해 수행되도록 하기 위해서, 각 Behavior를 활성화하는 시그널와 그 Behavior의 연산이 끝났음을 알리는 시그널을 Behavior 내에 삽입한다. 이 과정이 끝나면, Behavior내의 VHDL 코드를 그대로 출력 파일에 추가한다. 전이 아크는 loop문 속에서 앞서 생성한 신호를 wait문을 통해 처리하도록 함으로써 구현 가능하다. 만일 TI 아크가 포함되어 있는 경우, 예외 처리를 위해서 인터럽트 신호를 잡아낼 수 있는 기능을 추가한다. 모든 과정이 끝나면 그것을 다른 Behavior에게 알리는 신호를 발생시키고 블록을 닫으면, 하나의 Behavior가 VHDL로 변환된다. 각 과정은 재귀적으로 수행되므로 모든 Behavior가 VHDL로 변환되고 나면 SpecCharts로 기술된 전체 시스템은 동일한 기능의 VHDL로 표현되어 진다. 다음 4가지 항목은 SpecCharts를 Synopsys에서 합성 가능한 VHDL로 변환하는 SpecCharts 변환기를 구현하면서 사용한 기술적 특징을 정리한 것이다.

1) 계층적 모델 (Hierarchical Model)

SpecCharts의 상위 Behavior는 하위 Behavior를 활성/비활성 시킬 수 있어야 하며, 하위 Behavior는 상위 Behavior에게 자신의 동작이 완료되었음을 알릴 수 있어야 한다. 이러한 SpecCharts의 계층성을 표현하기 위해 VHDL의 중첩 블록(Nested Block)을 이용하였다. 각각의 Behavior는 Block문으로 만들고 Sub-behavior는 Sub-block으로 매핑한다. Leaf Behavior는 그 코드를 포함하는 단일 블럭으로 만든다. 각 Behavior의 선언은 각 Block의 선언이 된다.

2) 하위 Behavior의 제어

SpecCharts의 Behavior가 복합 Behavior일 경우 자신의 하위 Behavior를 제어하기 위한 방법이 필요하다. 이를 위해 Block의 마지막에 Control Process를 추가하였다. Control Process는 상위 Behavior로부터 활성 신호를 기다려, 그 신호가 오면 적절한 방법으로 하위 Behavior를 활성화시킨다. 하위 Behavior는 상위 Behavior로부터 비활성화 신호를 받으면, Leaf Behavior일 경우 실행되고 있는 코드를 중지시키고, 복합 Behavior일 경우 하위 Behavior 들을 비활성화 시킨다. 활성화되어 있는 동안 자신의 작업이 완료되면 complete 신호를 상위 Behavior로 보내서 자신이 작업을 완료하였음을 통지한다. 이 기능을 구현하기 위해 2개의 Boolean Signal(inBehavior, done-Behavior)을 사용하였다. inBehavior는 오직 상위 Behavior에 의해서만 제어되며, 자신의 하위 Behavior를 활성화/비활성화 시키는데 사용한다. done-Behavior는 하위 Behavior에 의해서 제어되며, 상위 Behavior에게 자신의 작업이 complete 되었음을 알리는데 사용한다.

3) Leaf Behavior의 처리

Leaf Behavior는 실제적인 연산 작업을 수행하므로 이 부분의 VHDL 변환에 가장 많은 주의를 기울였다. Leaf Behavior의 경우에는 Code Process문만을 추가하며, 하위 Behavior가 없으므로 Control Process는 추가하지 않는다. Leaf Behavior의 코드는 자신이 활성화/비활성화 되어 있는지에 따라 수행을 달리해야 한다. Code Process문 내에서 활성화 신호를 기다려 코드를 수행하고, 코드의 끝에서 부모에게 수행이 완료되었음을 알리는 complete 신호를

보내고, 비활성화 신호를 기다린다. 상위 Behavior가 비활성화되면 하위 Behavior는 모든 연산을 중단하고, 제어를 초기의 wait 문으로 돌려서 다시 자신이 활성화되기를 기다려야 한다. 따라서 Leaf Behavior가 비활성화되면 신호 할당 등의 모든 작업을 중단시켜 더 이상의 신호 발생을 막는다. Leaf Behavior가 또 다른 wait문을 수행하고 있는 동안 비활성화 신호를 받지 못하는 경우를 막기 위해서, 모든 wait문은 inBehavior 문장을 검사하여 false가 되면 중지하도록 설계하였다. TI 아크는 하위 Behavior의 실행을 즉시 중단시키는 일종의 인터럽트로써 예외 처리 기능을 가진다. 일반적으로 Leaf Behavior는 wait until 문을 사용해서 clock과 동기를 맞춘다. TI 아크가 연결된 경우에는 자신의 inBehavior가 false인지를 검사하는 기능을 수행하도록 해야 한다. 즉, wait문에 or not(inBehavior)문을 추가한다. not(inBehavior) 신호를 받으면 제어가 신호 갱신이 발생하지 않는 behavior의 제일 끝으로 옮겨가야만 한다. 그러나, VHDL에는 "go to"문이 없기 때문에 "labeled loop" 문장을 통해 이 문제를 해결한다. TOC 아크는 상위 Behavior에게 자신이 complete 되었음을 알리고 비활성화 신호를 기다리기만 하면 된다.

4) 변수의 처리

전역 변수는 같은 이름을 갖는 전역 signal로 바꾸었다. 그 변수에 값을 넣는 Leaf Behavior를 찾아서, 그 안에서 지역 signal을 선언하고, 그 값을 Leaf Behavior가 시작하는 당시의 전역 signal 값으로 초기화시킨다. 전역 변수에 값을 넣으면, 전역 signal을 갱신시킨다. Sequential Subbehavior는 한번에 하나의 Behavior만 활성화될 수 있으므로, 활성화되지 않은 나머지 하위 Behavior는 반드시 signal 에 null값을 할당해야만 한다. 그래서, 각 Leaf Behavior의 끝에서 각각의 signal에 null을 할당한다.

5. 실 험

여기에서는 ScSE를 구현하면서 테스트한 내용을 설명한다. 실험을 위한 웹-서버로는 Sun Enterprise 3000 머신에서 Netscape Enterprise Server 3.0을 이용하였으며, 웹-브라우저는 익스플로러 5.0을 사용하였다.

5.1 ScSE의 동작 검증

구현된 ScSE가 올바르게 동작함을 검증하기 위해 다음 5개의 예제에 대해 실험하였다.

① Counter : 클록에 의해 동기화되는 제어 기능을 갖춘 16진 업/다운 카운터.

② UART (Universal Asynchronous Receiver/Transmitter) : 패리티 비트를 포함하는 직렬(병렬) 8비트 입력을 병렬(직렬) 출력으로 내보내는 회로.

③ Pipelined Processor : fech, execute, write-results 단이 파이프라인(pipeline)되는 2-주소 명령 RISC 프로세서.

④ Answering Machine : 메시지 디스플레이, 녹음 등의 기능을 포함하는 자동응답기의 동작을 모델링한 회로.

⑤ Intel 8237 DMA(Direct Memory Access) Controller : 메모리와 직접 데이터를 주고, 받을 수 있는 외부 장치인 마이크로프로세서 시스템을 위한 주변 인터페이스 회로.

각 예제는 ScSE의 웹 인터페이스를 통해 로그인한 다음, SpecCharts 에디터 상에서 시스템을 기술하고, HW/SW 통합설계 응용 서버와의 통신으로 그 결과를 서버에 저장하도록 하였다. 저장된 SpecCharts 기술은 SpecCharts 변환기를 통해 VHDL로 변환하였다. 변환 결과로 생성된 VHDL 파일을 Synopsys를 이용해 시뮬레이션 함으로써, 시스템 설계자가 의도한대로 ScSE가 정확히 동작함을 확인하였다. 그림 10은 SpecCharts 에디터 상에서 자동 생성시킨 16진 업/다운 Counter의 텍스트 표현이다.

Counter는 2개의 sequential subbehavior(Count, Clear)를 가진다. Counter의 상태 변환은 제어 신호인 CON에 의해서 수행되는데, CON이 "00"일 경우 TI 아크에 의해서 카운터를 초기화하는 Clear로 상태가 전환되며, 그렇지 않을 경우에는 업/다운 카운터 기능을 수행하는 Count로 상태가 전환된다. Count는 3개의 sequential subbehavior(wait_state, Count_up, Count_down)로 구성되는데, CON이 "10"일 경우 업 카운트 기능을 수행하는 Count_up으로, CON이 "11"일 경우 다운 카운트 기능을 수행하는 Count_down으로 상태가 전환된다. 그림 11은 Synopsys 상에서 Counter 예제를 시뮬레이션한 결과를 보여준다. 시뮬레이션 결과의 분석을 통해, CON 신

```
use work.bit_functions.all;
entity ControlledCounterE is
port (
    CLK : in bit;
    CON : in bit_vector(1 downto 0);
    CNT_OUT : out bit_vector(3 downto 0)
);
end;
architecture ControlledCounterA of ControlledCounterE is
begin
    behavior Counter type sequential subbehaviors is
        signal CNT : integer;
        function rising(signal s : bit) return boolean is
        begin
                return (s = '1' and s'event);
                    end;
begin
        Count : (TI, CON = "00", Clear);
        Clear : (TI, not(CON = "00"), Count);
        behavior Count type sequential subbehaviors is
        begin
                wait_state : (TI, CON = "10" and rising
                            (CLK), Count_up),
                (TI, CON = "11" and rising(CLK), Count_
                            down);
        Count_up : (TOC, true, wait_state);
        Count_down : (TOC, true, wait_state);
        behavior wait_state type code is
        begin
                null;
        end wait_state;
        behavior Count_up type code is
        begin
                CNT <= CNT + 1;
                CNT_OUT <= I2B(CNT, 4);
        end Count_up;

        behavior Count_down type code is
        begin
                CNT <= CNT - 1;
                CNT_OUT <= I2B(CNT, 4);
        end Count_down;
        end Count;
        behavior Clear type code is
        begin
                CNT_OUT <= B"0000";
                CNT <= 0;
        end Clear;
    end Counter;
end ControlledCounterA;
```

그림 10. Counter 예제의 텍스트 표현

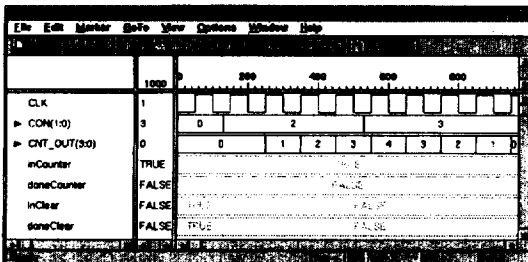



그림 11. Synopsys 시뮬레이션 결과

호가 0("00")일 경우 Clear 상태로 전환되며, 0이 아닐 경우 Count로 상태가 전환됨을 알 수 있었다. 또한 CON 신호가 2("10")인 경우에는 1씩 증가하는 업 카운트 기능을, 3("11")인 경우에는 1씩 감소하는 다운 카운트 기능을 수행해서, SpecCharts 에디터에서 기술한 의도대로 정확히 동작하고 있음을 확인하였다.

5.2 SpecCharts와 VHDL의 복잡도 비교

그림 12는 SpecCharts의 텍스트 표현을 이용한 코드와 SpecCharts 변환기를 통해 생성된 VHDL 코드를 라인 수로 비교한 것이다. 그림 12의 결과로 ScSE를 이용하면, VHDL을 이용했을 때보다 시스템 기술의 복잡도를 많이 감소한다는 것을 알 수 있다. ScSE를 이용하지 않고 VHDL로 직접 시스템을 설계하게 되면 생성된 VHDL과 같이 3배 이상의 코드를 직접 코딩해야만 한다.

또한 ScSE를 이용하는 시스템 설계자는 SpecCharts의 그래픽 표현을 이용해서 설계하기 때문에, 예외 처리나 모듈의 계층성과 같은 사항을 시각적으로 점검하며 설계에 임할 수 있다. 그것은 ScSE가 시스템 설계자로 하여금 보다 상위 수준에서 시스템 설계를 할 수 있도록 유도하고 있음을 보여주는 것이다.

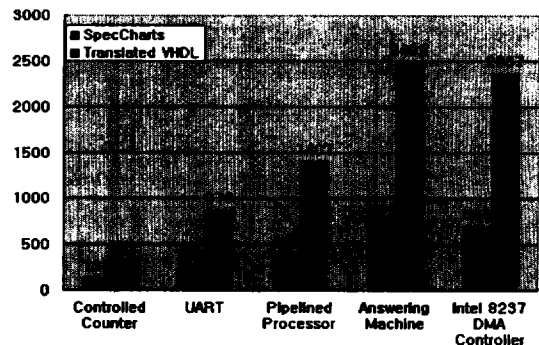



그림 12. SpecCharts와 생성된 VHDL 라인 수 비교

5.3 Xspeccharts와의 비교

SpecSyn은 SpecCharts를 개발한 U.C. Irvine에서 개발 중인 SpecCharts 전용 통합설계 도구이다. 그림 13은 SpecSyn에서 시스템을 기술할 때 사용되는 Xspeccharts이다. 본 연구에서 개발한 SpecCharts 에디터는 Xspeccharts와 비교할 때 다음과 같은 장

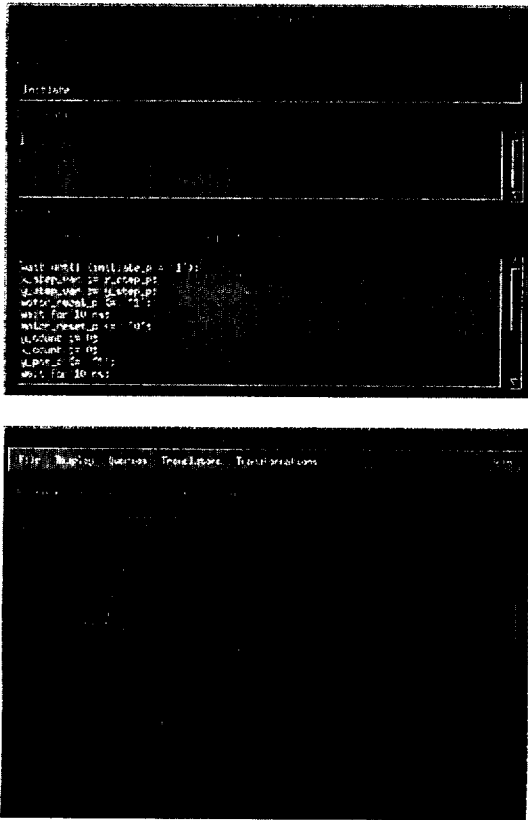



그림 13. SpecSyn의 Xspeccharts의 기술 환경

점을 가진다.

Xspeccharts는 전체 Behavior의 계층적 표현에 있어서 전이 아크를 도시하지 않음으로써 설계자로 하여금 상태 전환 시점을 알기 어렵게 한다. 또한 한정된 설계 영역으로 인해 대형 시스템 설계시 불편하다. 또한 똑같은 크기와 동일한 색으로 Behavior를 표현함으로써 Behavior의 종류를 알기 위해서는 각 Behavior의 코드를 직접 실행시켜야 하는 불편함이 있다. 이와 비교해서 SpecCharts 에디터는 Java를 이용해서 다중 플랫폼 환경에서 동작된다. 또한 전이 아크와 전이가 발생하는 조건을 명확히 보여줌으로써 상태 전환 시점을 직관적으로 알 수 있도록 설계되었다. 시스템이 복잡해지면 각 Behavior를 새로운 윈도우를 띄워 독자적으로 설계할 수 있는 다이브(Dive) 기능을 추가하여 대형 시스템 설계시 설계 공간 부족 현상을 없앴다. 또한 각 Behavior 종류에 따라 서로 다른 방법으로 표기함으로써 시스템 설계자가 설계 도중에 겪을 수 있는 설계 혼동을 최소화하였다.

6. 결 론

본 논문에서는 기존의 HW/SW 통합설계 도구가 가지는 플랫폼 의존성, 잦은 기술 변화와 그에 따른 잦은 설치 및 업그레이드의 불편함, 익숙하지 못한 인터페이스, 협동 작업에 대한 고려 부족 등의 단점을 해결하기 위해, 분산 구조를 갖는 웹-기반 HW/SW 통합설계 환경인 WebCEDA를 제안하였다. 또한 WebCEDA에서 시스템을 기술하기 위한 SpecCharts 기술 환경을 구현하였다. WebCEDA는 클라이언트와 HW/SW 통합설계 서버, 리모트 서버로 이루어진 3층 클라이언트/서버 구조를 가지며, 협동 작업 환경, 객체 지향 사용자 인터페이스, 트랜잭션 관리, 보안 등을 지원하도록 설계되었다. ScSE는 VHDL을 확장한 SpecCharts를 시스템 기술 언어로 채택해 시스템 설계자의 학습 부담을 최소화하였으며, SpecCharts 변환기를 구현함으로써 Synopsys, VDT와 같은 VHDL을 위한 기존의 설계 환경을 재사용할 수 있도록 하였다.

SpecCharts 에디터는 SpecSyn의 시스템 기술 환경인 Xspeccharts와 비교할 때, 플랫폼에 독립적인 사용 환경과 대형 시스템 설계시의 겪게 되는 설계 공간 부족 문제의 해결, 직관적인 인터페이스를 통한 설계 혼동의 최소화와 같은 장점을 가진다. 웹 상의 SpecCharts 에디터를 통해 기술된 Counter 예제를 SpecCharts 변환기를 통해 VHDL로 변환하고, 그 결과를 Synopsys 상에서 시뮬레이션 함으로써 ScSE의 논리적인 정확성을 검증할 수 있었다. 또한 SpecCharts 기술과 SpecCharts 변환기를 통해서 변환된 VHDL의 라인 수를 비교함으로써, ScSE를 이용할 경우 기존의 시스템 설계 방법에 비해 시스템을 기술의 복잡성을 최대 75%까지 감소시킬 수 있음을 확인하였다.

현재 본 연구의 결과를 토대로 WebCEDA의 전체적인 프레임웍을 구현 중에 있다. 추후 연구과제로 HW/SW 통합설계 분야의 진보된 기술 변화를 프레임웍에 보다 쉽게 포함시킬 수 있도록 하기 위해서 CORBA와 같은 분산 객체 기술을 이용해서 전체적인 프레임웍을 확장하는 것이 필요하다고 본다. 또한

WebCEDA가 널리 사용되기 위해서는 적합한 보안 모델을 설계하고 구현하는 일이 선행되어야 한다.

참 고 문 헌

[1] Jerzy Rozenblit, Klaus Buchenrieder, "Codesign : Computer-Aided Software/Hardware Engineering"", IEEE PRESS, 1995.

[2] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A, Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, B. Tabbara, "Hardware-Software Co-Design of Embedded Systems: The Polis Approach", Kluwer Academic Press, 1997.

[3] R. Niemann, P. Marwedel, "Synthesis of Communicating Controllers for Concurrent Hardware/Software Systems", Design, Automation and Test in Europe (DATE), 1998.

[4] H. Oh and S. Ha, "A Hardware/Software Cosynthesis Technique Based on Heterogeneous Multiprocessor Scheduling," 7th International Workshop on Hardware/Software CO-Design, Rome, Italy, May 1999.

[5] R. Ortega, G. Borriello, "Communication Synthesis for Embedded Systems with Global Considerations", Proc. Codes/CACHE, pp. 69-73, 1997.

[6] R. Esser, J. Teich, L. Thiele, "CodeSign: An Embedded System Design Environment", J. Proc. of the IEEE, Computers and Digital Techniques, pp. 171-180, 1998.

[7] D. Gajski, J. Gong, F. Vahid, S. Narayan, "The SpecSyn Design Process and Human Interface," UC Irvine, Technical Report ICS-TR-93-03, 1993.

[8] Linda Geppert, "IC Design on the world wide web", IEEE Spectrum, p. 45-50, April-June, 1998.

[9] M. D. Spiller, A. R. Newton, "EDA and the Network", Proc. of the IEEE International Conference on Computer-Aided Design, pp. 470-476, Nov, 1997.

[10] S. Yoo, K. Choi, "Optimistic Distributed Timed Cosimulation Based on Thread Simulation Model", Proc. Int'l Workshop on Hardware/Software Co-Design(Codes/CASH E), Mar. 1998.

[11] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy : A Framework for Simulating and Prototyping Heterogeneous Systems", International Journal of Computer Simulation, Vol 4, pp. 155-182, April, 1994.

[12] G. Berry, "A hardware Implementation of pure Esterel", Digital Equipment Paris Research Laboratory, July, 1991.

[13] IOS IS 8807, Information Processing Systems, Open System Interconnection, LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behavior, IOS, July, 1988.

[14] S. Narayan, F. Vahid, and D. D. Gajski. "System specification with the SpecCharts language". In IEEE Design & Test of Computers, Dec. 1992.

[15] 김승권, 김종훈, "HW/SW 통합설계를 위한 SVT (SpecCharts_to_VHDL Translator)의 설계 기법", 한국정보과학회 가을학술발표논문집, 24권, 2호, pp. 741-744, 1997.

[16] http://poppy.snu.ac.kr/VDT, VHDL Developer's Toolkit (VDT)

[17] 김승권, 박종호, 김종훈, "실시간 시스템의 HW/SW 통합설계를 위한 확장된 Greedy 알고리즘", 한국정보과학회 봄학술발표논문집, 26권, 1호, pp. 717-719, 1999.

[18] Y. Kim, Y. Shin, K. Kim, J. Won, K. Choi, "Efficient prototyping system based on incremental design and module-by-module verification", Proc. of 1995 International Symposium on Circuits and Systems, pp. 924-927, May, 1995.

김 승 권

1997년 동아대학교 컴퓨터공학과 졸업(공학사)
1999년 동아대학교 컴퓨터공학과 졸업(공학석사)
1999년~현재 동아대학교 컴퓨터공학과(박사과정)
관심분야 : HW/SW 통합설계, 보안 프로토콜, 실시간 시스템

김 종 훈

1974년 동아대학교 전자공학과 졸업(공학석사)
1977년 동아대학교 전자공학과 졸업(공학석사)
1986년 경북대학교 전자공학과 졸업(공학박사)
1986년~현재 동아대학교 컴퓨터공학과 교수
관심분야 : HW/SW 통합설계, 실시간 시스템, 정보 보안